Stanford CS229 Project: Travel Landmark Image Classification using Varied ML Techniques

Project Category: Computer Vision

Michelle Xing (mxing621), Hanna Yip (hannayip), Trinity Donohugh (tringd)
Department of Computer Science, Stanford University
suid@stanford.edu

1 Introduction

The task for this project is to classify different landmarks in images (i.e. the Colosseum, the Eiffel Tower, the Empire State Building). The problem that we address is not being able to remember or recognize a landmark in a photo. Since the advent of the cloud, photo collections have increasingly grown in size and people are able to record more and more memories. However, the photos containing landmarks (which people have taken during their past travels), are often forgotten as to where they are situated and what they are called. Consequently, we plan to apply many different ML models to a subset of the Google-Landmarks dataset in this project [2]. The input to our ML algorithms is a 2D image with 3 channels (red, green, blue). We then use various feature extraction methods such as mean of pixel channels and edge detection to obtain feature representations of the images. Finally we use four ML classifiers – Support Vector Machine, Logistic Regression, K Nearest Neighbors, and Random Forest – to perform predictions.

This is a very interesting problem to pursue in the machine learning space besides its practical applications particularly because with the development of deep learning, an increasingly large portion of computer vision tasks use CNNs to extract out desired, latent features in images which allows these algorithms to achieve high accuracies on their tasks. It is expected that the combination of feature extraction techniques and machine learning models will not achieve the same level of accuracy as the top leaderboard models on the Kaggle Google Landmark Recognition Challenge. However, exploring the space of feature extraction and traditional ML models offers many pedagogical learnings as detailed in later sections.

2 Related work

Instance Image retrieval. Traditionally, image retrieval is the problem of finding similar images in embeddings with small distances between them using local descriptor based methods, such as the popular SIFT [7], RootSIFT [8], and SURF [9]. Recently, methods such as Bag-of-words model [10] and it's variants VLAD [11] and Fisher Vector [12] provide a variation to the traditional methods by constructing image embedding with an aggregation of local descriptors. More recently, Google proved that DELF [13] can outperform traditional models by using the attention map of the CNN activation layer learned by only image-level annotations. The Two-stage Discriminative Re-ranking for Large-scale Landmark Retrieval [14] paper also showed improved results by using traditional CNN-based methods to learn the embedding but then the label information to perform a two-step re-ranking process which uses a discriminative model based on k-NN search with soft voting allowing initially retrieved results to be sorted and then adds originally not retrieved images into the retrieval results based on the same discriminative model. This enabled the model to be able to recognise images of both the inside and outside of the landmark. An additional approach channeled by the Deep Image Retrieval: Learning global representations for image search paper [15] showed that they could produce a global and compact fixed-length representation for each image by aggregating many region-wise descriptors, avoiding the black-box problem of most CNN-based feature extractors. In our project, we used similar methods of finding similar distances in embedding space by using the k-nearest neighbours algorithm. We also experimented with random forest as we saw it as a similar method to re-ranking where the optimal probability prediction is chosen at every time-step, logistic regression, and svm. Since we wanted to limit out project to traditional machine learning methods,

we could not use CNN-based methods for feature extraction and instead used mean of channels, histogram of oriented gradients (HOG), vertical and horizontal Prewitt filters, Roberts filter, and Canny edge detector. An upside to this is that we know exactly which feature extraction methods are used and avoid the black-box problem.

Instance-Level Recognition and Retrieval datasets. Image classification problems range from basic high-level categorizations to distinctions based on style, model or species, and instance-level recognition that can tell you the exact human-given name of the objects/ shapes image. In our project, we aim to do classification using an instance-level recognition and retrieval dataset inspired by the Google Landmarks Dataset v2 [2]. This dataset contains over 5M images of over 200k human-made and nature landmarks. Other datasets the Places dataset [6] that contain over 10 million images of scene photographs. Although, this dataset is only labeled with high-level scene semantic categories.

3 Dataset and Features

3.1 Dataset

	id	url	landmark_id
0	97c0a12e07ae8dd5	http://lh4.ggpht.com/-f8xYA5l4apw/RSziSQVaABI/...	6347
1	650c989dd3493748	https://lh5.googleusercontent.com/-PUnMrX7oOyA...	12519
2	05e63ca9b2cde1f4	http://mw2.google.com/mw-panoramio/photos/medi...	264
3	08672eddcb2b7c93	http://lh3.ggpht.com/-9fgSxDYwhHA/SMvGEoltKTI/...	13287
4	fc49cb32ef7f1e89	http://lh6.ggpht.com/-UGAXxvPbr98/S-jGZbyMIPI/...	4018

Figure 1: CSV file data format

We downloaded our dataset from the Google Landmark Recognition Challenge on Kaggle. We used the test.csv and train.csv files to have a 80-20 train to test set split. Due to time constraints we choose not to have a validation set but look to add that in the future for further model fine tuning. The dataset came in the form of image id, url link to the image, and landmark id as we can see in Figure 1.

Figure 2: Images grouped into landmark classes of six images for each landmark

In the data pre-processing step, we first sampled the data since we were using numpy to train our model which limits the size of our dataset and we wanted to remove non-landmark images in the dataset. Thus, we produced a data sample of 2000 classes with images of resolution 96x64 inspired by the techniques used by Anisha Garg [5]. Then, since the data came in CSVs as seen above with image ids, URLs, and landmark ids, we split the sampled dataset into 16,000 images for training and 4,000 for testing and ran a loop through the CSV file to download the images from every link, catching the links that have either been deprecated or no longer work along the way, and splitting the images into a train and test folder. Finally, within the train folder we split the data into specific directories with each directory corresponding to a landmark class. The images that returned in each class can be seen in Figure 2, where each class contains images of the same landmark from different angles, lighting, and seasons.

3.2 Features

We extracted features from the images using a variety of methods including mean of channels, histogram of oriented gradients (HOG), vertical and horizontal Prewitt filters, Roberts filter, and Canny edge detector. Examples of a resized image after each of the feature extractors is shown in Figure 3.

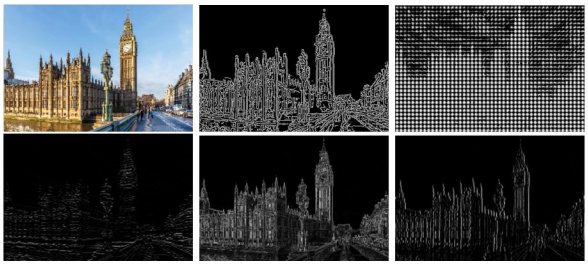

Figure 3: Clockwise, we have the original resized image, then the image produced by Canny, HOG, horizontal Prewitt, Roberts, and vertical Prewitt

For mean of channels, we produce 1 feature for each pixel by averaging over the values in the RGB channels, a method more complex and effective than just using the black and white values for the pixels. HOG, popular for object detection, works by normalizing the illumination of an image, then computing the first order image gradients which capture color, silhouette, and texture. Finally, HOG computes gradient histograms which are encodings that are affected by image content locally, but resists overall small discrepancies. Vertical and Horizontal Prewitt captures the vertical and horizontal edges of an image by using the Prewitt transform and their standard kernels. The Roberts filter is also an edge detector and finds the edge magnitude by using Roberts' cross operator. Finally, the Canny edge detector is a robust algorithm that finds edges and can be approximated by the first derivative of the Gaussian. Besides a Gaussian filter, the algorithm finds the intensity gradients in the image and applies thresholding for edge detection. Note that the Prewitt filters, the Roberts filter, and the Canny edge detector all first transform the 3-channel image into a black and white image.

4 Methods

In this project, we experimented with four machine learning methods to see which method performed best on this classification task: logistic regression, SVM, K Nearest Neighbors, and Random Forest. Before running our machine learning models on the actual Google-Landmarks dataset, we honed our feature extraction methods using a small, toy dataset of 100 images that were divided equally among 10 classes of landmarks that are also in the Google-Landmarks dataset. Similar to how the Google-Landmarks dataset was constructed, different views and difference parts of the landmarks were incorporated into the variety of images. We, as human annotators, established the ground truth correspondences between the images and the labels. Once we figured out the most useful features to extract from images for this classification task, we ran the various ML models on the real dataset and then finetuned the best algorithm for tackling our specific problem. Below is a description of each of the machine learning methods we experimented with.

Logistic Regression

Logistic regression is a supervised machine learning method that uses a linear regression equation to produce discrete set of classes. The cost function is defined as a sigmoid function which maps predicted values to probabilities. The objective function performs maximum likelihood estimation (MLE) by maximizing the log likelihood. Using gradient descent, we find the parameters that minimize the cost value. Below is the likelihood equation:

$$\begin{aligned} \ell(\theta) &= \log L(\theta) \\ &= \sum_{i=1}^{n} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)})) \end{aligned}$$

Support Vector Machine

SVM is another supervised machine learning algorithm used for classification tasks. SVM iteratively generates hyperplanes which segregate the dataset into different classes. It then selects the maximum marginal hyperplane which is the hyperplane with the maximum segregation from nearby data points. In non-linear spaces where data cannot be separated using a linear hyperplane, SVM uses a kernel trick to transform the input space to higher dimensional space. The optimization problem is as follows:

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha_i \left[y^{(i)}(w^T x^{(i)} + b) - 1 \right]$$

K Nearest Neighbors

K Nearest Neighbors is a type of non-generalizing learning. It does not construct a general internal model, but simply stores instances of training data and classifies a new data point based on the similarity. Classification is computed from a majority vote of the nearest neighbors of each point: a query point is assigned to the class with the most representatives within the nearest neighbors of the point. It assumes the similarity between the unseen test data and stored, available train data.

Random Forest

Random Forest fits a number of uncorrelated decision tree classifiers on various subsamples of the dataset and averages across their predictions to improve prediction accuracy and control over-fitting. Since they are uncorrelated, trees protect each other from their individual errors. Random Forest also adds additional randomness to the model - instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features.

5 Experiments/Results/Discussion

Experiments. The metrics we used to evaluate the performance of the models include: accuracy, precision, recall, and F1 score. Accuracy is the ratio of correctly predicted observation to the total observations. Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. Recall is the ratio of correctly predicted positive observations to the all observations in actual class. F1 Score is the weighted average of Precision and Recall. We also visualized the performance of our models using a confusion matrix which explicates the false negatives and false positives of the classification task, and do a qualitative analysis.

For each of the classifiers, we performed cross validation with 5 folds to tune their hyperparameters. For example, for the SVM model, we investigated the performance of the model with different hyperparameters, including the kernel (the function that transforms the input data into the required form), the regularization (the penalty parameter which controls the trade-off between decision boundary and misclassification terms), and gamma (controls under and overfitting).

On our subsection of the Google Landmark Recognition Dataset, we first extracted features on both on the train and test sets. Then we took those feature representations of our dataset and ran each of the 4 ML models on them. The resulting accuracy matrix on the test set is shown in Table 1.

Results. The confusion matrices (Figure 4) of all the models qualitatively reflect how difficult this classification task is, especially since there are 2000 classes and pictures of landmarks are often just large structures positioned in front of a backdrop of a sky. Most of them show low accuracy on the diagonal, which indicates the model is not great at classifying landmarks correctly. However, there are some sparse points of yellow which indicates there are some classes the model gets right.

	canny	hog	hprewitt	mean	roberts
K Nearest Neighbors	-	11%	3%	6%	-
Logistic Regression	-	12%	5%	7%	-
Random Forest	-	4%	3%	4%	-
SVC	6%	13%	6%	9%	8%

Table 1: Feature extractor and ML model accuracy matrix on test set. Blank entries due to python encountering a memory overload issue (Canny edge detector and the Roberts filter requires higher computational power)

Discussion. As you can see in Table 1, the various ML models were generally comparable given a certain feature extractor. However, HOG as a feature extractor achieved the best accuracies on the test set. This may be because HOG as shown Figure X is able to hone in on the landmark object in the photo whereas the other edge extractors such as Prewitt, Roberts, and Canny essentially outline anything in the image that has a shape, including people and other landscaping objects such as trees which may confuse the ML models if these distracting objects exist in images of any class. Qualitatively, we saw that for an image labeled with Eiffel Tower, only the models using HOG were

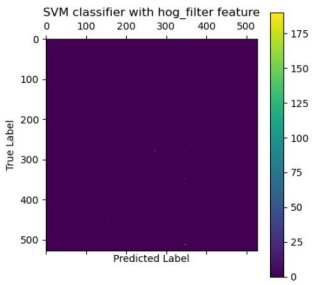

Figure 4: Confusion matrix of the SVM model trained using the HOG filter

Figure 5: From left to right, we have the original resized image containing the Eiffel Tower with multiple people in the frame, then the image produced by HOG, Canny, horizontal Prewitt, Roberts, and vertical Prewitt. This image was incorrectly labeled by all ML models that ingested feature vectors from extractors other than HOG.

able to correctly classify it. Looking at Figure 5, we see that the Eiffel Tower is not blocked in the photo by other objects; however, the landmark isn't centered and there are 3 people in the photo nearly as large as the landmark to detect. Thus, when we run the feature extractors Prewitt, Roberts, and Canny on the image, we see that all 3 people are captured clearly. Meanwhile, as mentioned in section 3.2, HOG resists overall small discrepancies and recognizes that the bottom half of the photo contains a lot of noise.

In regards to related works for this particular problem, deep neural networks in this computer vision space do well because earlier layers detect edges and later layers abstract out larger objects and features. However, using solely ML which is mainly able to detect edges for features captures local features of the image but not features on a larger scale. Thus, there is no way of deriving the relationship between pixels far away from each other i.e. in an image of a landmark building, knowing that the top right point of the building is meant to eventually meet the bottom left point of the building by means of local edges.

Another interesting point is that the feature extractors Prewitt, Roberts, and Canny all first convert the image to black and white whilst mean and HOG captures color. The weather and lighting and hues of the image can be different, but certain landmarks such as the Statue of Liberty have very distinct colors, so capturing color could've helped with accuracy.

6 Conclusion/Future Work

The Support Vector Classifier with the HOG feature extractor was the highest performing. We think the HOG feature extractor was the most successful because given the Google Landmarks Dataset containing images of landmarks from different angles, with people and other objects in them, it was able to disregard those distractions. Overall, given a specific feature extractor, the various ML models achieved comparable accuracies. This highlights the idea that in traditional ML, feature mapping is incredibly paramount.

If we had more computational resources and time, we would love to work on the full Google Landmarks dataset which has > 5 million images spanning 200 thousand classes. During this project, we also ran into problems with using non GPU libraries by using Scikit Learn and would love implement these algorithms in PyTorch. For further exploration, we would love to see just what the upper bound of the accuracy of traditional ML techniques on the Google Landmark Dataset Challenge would be.

7 Contributions

We worked in parallel and contributed equally to the project idea, models, experiments, and paper.

8 References

[1] H. Noh, A. Araujo, J. Sim, T. Weyand, B. Han, "Large-Scale Image Retrieval with Attentive Deep Local Features", Proc. ICCV'17

[2] "Google Landmarks Dataset v2 - A Large-Scale Benchmark for Instance-Level Recognition and Retrieval" T. Weyand*, A. Araujo*, B. Cao, J. Sim Proc. CVPR'20

[3] "Detect-to-Retrieve: Efficient Regional Aggregation for Image Search", M. Teichmann*, A. Araujo*, M. Zhu and J. Sim, Proc. CVPR'19

[4] "Unifying Deep Local and Global Features for Image Search", B. Cao*, A. Araujo* and J. Sim, Proc. ECCV'20

[5] https://towardsdatascience.com/google-landmark-recognition-using-transfer-learning-dde35cc760e1

[6] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva and A. Torralba, "Places: A 10 Million Image Database for Scene Recognition," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 6, pp. 1452-1464, 1 June 2018, doi: 10.1109/TPAMI.2017.2723009.

[7] David G. Lowe. Distinctive image features from scaleinvariant keypoints. IJCV, 60(2):91–110, 2004.

[8] Relja Arandjelovic and Andrew Zisserman. Three things everyone should know to improve object retrieval. In CVPR, pages 2911–2918, 2012

[9] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: speeded up robust features. In ECCV, pages 404–417, 2006.

[10] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In ICCV, pages 1470–1477, 2003

[11] Herve Jegou, Florent Perronnin, Matthijs Douze, Jorge Sanchez, Patrick Perez, and Cordelia Schmid. Aggregat-ing local image descriptors into compact codes. TPAMI, 34(9):1704–1716, 2012

[12] Florent Perronnin, Yan Liu, Jorge Sanchez, and Herve Poirier. Large-scale image retrieval with compressed fisher vectors. In CVPR, pages 3384–3391, 2010.

[13] Hyeonwoo Noh and Andre Araujo and Jack Sim and Tobias Weyand and Bohyung Ha. Large-Scale Image Retrieval with Attentive Deep Local Features. 2018.

[14] Yokoo, Shuhei, et al. "Two-stage discriminative re-ranking for large-scale landmark retrieval." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. 2020.

[15] Gordo, Albert, et al. "Deep image retrieval: Learning global representations for image search." European conference on computer vision. Springer, Cham, 2016.